

This Page Is Inserted by IFW Operations
and is not a part of the Official Record

BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images may include (but are not limited to):

- BLACK BORDERS
- TEXT CUT OFF AT TOP, BOTTOM OR SIDES
- FADED TEXT
- ILLEGIBLE TEXT
- SKEWED/SLANTED IMAGES
- COLORED PHOTOS
- BLACK OR VERY BLACK AND WHITE DARK PHOTOS
- GRAY SCALE DOCUMENTS

IMAGES ARE BEST AVAILABLE COPY.

As rescanning documents *will not* correct images,
Please do not report the images to the
Image Problem Mailbox.

(19)



JAPANESE PATENT OFFICE

PATENT ABSTRACTS OF JAPAN

(11) Publication number: **01156824 A**

(43) Date of publication of application: **20.06.89**

(51) Int. Cl

**G06F 9/30
G06F 7/00
G06F 9/32
G06F 15/72**

(21) Application number: **62314063**

(22) Date of filing: **14.12.87**

(71) Applicant: **HITACHI LTD HITACHI MICRO
COMPUT ENG LTD SAKAMURA
TAKESHI**

(72) Inventor: **SAKAMURA TAKESHI
KAWASAKI IKUYA
HASEGAWA ATSUSHI
IWASAKI KAZUHIKO**

(54) MICROPROCESSOR

(57) Abstract:

PURPOSE: To secure the program flexibility and to facilitate the development of a program for the graphic processing, for example, by storing the information obtained by execution of a 1st instruction to an information holding means and controlling an instruction executing means based on said stored information when a 2nd instruction is carried out.

CONSTITUTION: The type of an arithmetic operation is defined as one of operands. In other words, a desired arithmetic operation is carried out by an instruction that is added the operand information designating the

type of the arithmetic operation to the outside or the inside of an operation designating part containing a common operation code showing an arithmetic operation (wide sense). Thus the contents of an operand are set previously to designate the type of the arithmetic operation based on the executing result of a certain instruction. Then the arithmetic operation is carried out based on the operand contents by the next instruction. Thus the type of the arithmetic operation can be changed into a dynamic one in a program. In such a way, the flexibility is secured to a program and the development of a program for graphic processing, for example, is facilitated.

COPYRIGHT: (C)1989,JPO&Japio

⑩ 日本国特許庁 (JP)

⑪ 特許出願公開

⑫ 公開特許公報 (A)

平1-156824

⑬ Int.Cl.

G 06 F 9/30
7/00
9/32

識別記号

3 4 0
3 5 0

厅内整理番号

A-7361-5B
B-7313-5B
B-7361-5B

⑭ 公開 平成1年(1989)6月20日

⑮ 発明の名称 マイクロプロセッサ

⑯ 特願 昭62-314063

⑰ 出願 昭62(1987)12月14日

⑱ 発明者 坂村 健 東京都港区白金台3-12-30-105

⑲ 発明者 川崎 郁也 東京都小平市上水本町1450番地 株式会社日立製作所武藏工場内

⑳ 出願人 株式会社日立製作所 東京都千代田区神田駿河台4丁目6番地

㉑ 出願人 日立マイクロコンピュータエンジニアリング 東京都小平市上水本町1479番地

株式会社

㉒ 出願人 坂村 健 東京都港区白金台3-12-30-105

㉓ 代理人 弁理士 小川 勝男 外1名

最終頁に統く

明細書

1. 発明の名称

マイクロプロセッサ

2. 特許請求の範囲

1. 命令解説手段と、
命令実行手段と、
情報保持手段とを含み、

第1の命令の実行によって得られた情報を上記情報保持手段に格納する第1のステップと、
第2の命令の実行の際に上記情報に基づいて上記命令実行手段が制御される第2のステップを含むことを特徴とするマイクロプロセッサ。

2. 上記命令実行手段は論理演算ユニットを含み、
上記情報に基づいて、上記論理演算ユニットの演算機能が制御されることを特徴とする特許請求の範囲第1項記載のマイクロプロセッサ。

3. 上記情報保持手段は汎用レジスタであること
を特徴とする特許請求の範囲第2項記載のマイクロプロセッサ。

4. 命令解説手段と、

命令実行手段とを含み、

上記命令解説手段によって解説される命令のオペコード内には、替替え可能な情報保持手段に記憶された情報に基づいて上記命令実行手段の少なくとも一部の制御を実行させる情報を含むことを特徴とするマイクロプロセッサ。

5. 上記情報が記憶される上記情報保持手段内の番地を指定するための番地情報は、上記オペコード内に含まれていることを特徴とする特許請求の範囲第4項記載のマイクロプロセッサ。

6. 上記情報が記憶される上記情報保持手段内の番地を指定するための番地情報は、オペランド領域に含まれていることを特徴とする特許請求の範囲第4項記載のマイクロプロセッサ。

7. 上記替替え可能な情報保持手段とは、マイクロプロセッサ内のレジスタであることを特徴とする特許請求の範囲第4項記載のマイクロプロセッサ。

8. 上記命令実行手段は論理演算ユニットを含み、
上記レジスタの内容に基づいて上記論理演算ニ

ニットの演算機能が制御されることを特徴とする特許請求の範囲第7項記載のマイクロプロセッサ。

9. 上記レジスタの内容に基づいて上記論理演算ユニットの演算の種類が選択されることを特徴とする特許請求の範囲第8項記載のマイクロプロセッサ。

10. 上記命令解説手段によって解説される命令は、メモリ内の任意のビットから任意のビットまでの領域のデータの取り扱いに関する命令であることを特徴とする特許請求の範囲第9項記載のマイクロプロセッサ。

3. 発明の詳細を説明

(産業上の利用分野)

この発明は、データ処理技術さらにはプログラム制御方式のシステムにおける命令体系に適用して特に有効な技術に関し、例えばビットフィールドと呼ばれるデータの取り扱いに関する命令を有するマイクロプロセッサを利用して有効な技術に関する。

例えば、コンピュータ・グラフィックのような分野においては、ビットフィールド内のデータに対し論理演算を施して、いわゆる塗りつぶしやすかしなどの描画処理を行なわせる場合、画面を見ながら演算の種類をダイナミックに決定することができればプログラムの開発が容易となる。

しかるに、演算の種類によって命令が決っている従来のマイクロプロセッサでは、演算処理の内容を変えるにはプログラムの中の演算命令を書き換えなくてはならず、プログラムに柔軟性がないという不都合があった。

また、ある命令の実行によって得られた結果に基づいて次の命令又は演算の種類を決定するためには、プログラム上、次に実行する可能性のある命令又は演算を羅列しておかなければならぬ。すなわち、上記ある命令の実行によって得られた結果に基づいて上記羅列された命令等の1つを選択する様なプログラムを作成しなければならない。従ってプログラムの柔軟性がないだけでなく、命令を選択する等の処理を必要とするから、一連の

(従来の技術)

従来、マイクロプロセッサには、加算、減算、乗算、除算、比較などの算術演算命令の他、論理積(AND)、論理和(OR)、排他的論理和(XOR)など種々の論理演算命令が備えられている。例えば(株)日立製作所、1982年9月発行、「日立マイクロコンピュータ、SEMICONDUCTER DATA Book、8/16ビットマイクロコンピュータ」p914~p919, p945~p952等に記載されている。

(発明が解決しようとする問題点)

従来のマイクロプロセッサにおける命令体系では、演算の種類は命令(オペレーションコード)で指定するようにされていた。つまり、各演算ごとに命令が用意されており、演算の種類はプログラム上において固定されてしまい、データのように変更することはできなかった。従って、プログラムがROM(リード・オンリ・メモリ)内に格納された場合、演算を変更することは不可能であった。

命令を実行する際の高速動作が制限されてしまう。

この発明の目的は、マイクロコンピュータ・システムにおけるプログラムに柔軟性を持たせ、例えばグラフィック処理用のプログラムの開発が容易に行なえるような演算命令に関する命令形式を提供することにある。

この発明の前記ならびにそのほかの目的と新規な特徴については、本明細書の記述および添付図面から明らかになるであろう。

(問題を解決するための手段)

本願において開示される発明のうち代役的なものの概要を説明すれば、下記のとおりである。

すなわち、演算の種類をオペランドの一つとして与える、つまり演算(広義)といいう共通のオペレーションコードの入ったオペレーション指定部の外又は中に、演算の種類を指定するオペランド情報を附加した命令によって所望の演算を実行させるようとするものである。

(作用)

上記した手段によれば、ある命令の実行結果に

基づいて、演算の種類を指定するオペラントの内容を設定しておき、次の命令で上記オペラントの内容に従って演算できるから、プログラムの中で演算の種類をダイナミックに変えることができる。従ってプログラムに柔軟性を持たせ、例えばグラフィック処理用のプログラムの開発が容易に行なえるようにするという上記目的を達成することができる。

以下、本発明を一例として、ビットフィールドと呼ばれるメモリ内の任意のビットから任意のビットまでのデータの取り扱いに関する命令（以下ビットフィールド命令と称する）に適用した場合の一実施例を説明する。

〔実施例〕

ビットフィールド命令は、第1図及び第2図に示すようにベースアドレスBAとこのベースアドレスからのオフセットOff[およびフィールド長さ（ビット数）を示すフィールド幅WDの3つの値をオペランドとして与えることによりメモリ内の所要のフィールドを指定し、そのフィールドの

演算の種類を示すコードは、予めMOVE命令等によりメモリ内からデータとして読み出して所定のレジスタR5に入れておく。

また、同様にベースアドレスBA、オフセットOff[及びフィールド幅WDもそれぞれ所定のレジスタ内に入っている値を使って命令を実行するようになっている。

第9図(A)又は第9図(B)に示す命令は、例えば、ビットフィールド命令であり、あるビットフィールド（ソース側）のデータと他のビットフィールド（デスティネーション側）のデータとの論理をとってそれをデスティネーション側のビットフィールドへ入れるというビットフィールド間演算命令である。この命令を実行するためには、ソース側のビットフィールドを特定するベースアドレスBA_s、オフセットOff_s及びフィールド幅WD_sをそれぞれ入れるレジスタと、デスティネーション側ビットフィールドを特定するベースアドレスBA_d、オフセットOff_d及びフィールド幅WD_dをそれぞれ入れるレジスタと、演算の

データに対してアンド(AND)やオア(OR)などの論理演算処理を行なうものである。なお、このようなビットフィールド命令は、例えばモトローラ社製MC68020のようないマイクロプロセッサで既に用意されている。このビットフィールド命令は、オペレーションコードの後のオペラントによってベースアドレスBAやオフセットOff[およびフィールド幅WDが与えられていた。

この実施例では演算の種類もオペランドで指定するものである。オペランドによる演算の指定の具体的な方法として、例えば第9図(A)の実施例では、レジスタ番号を有するレジスタ直後アドレスシング方式を用いた。すなわち、予め所定のレジスタR8内に演算の種類を示すコードを入れておく、オペランドにはそのコードの入ったレジスタ番号とアドレスシングモードを入れておくものである。第9図(B)の実施例では、オペレーションコード内に、レジスタR5の内容に基づいて演算の種類を定める旨の情報が付加されている。第9図(A)又は、第9図(B)に示す命令を実行する場合には、

種類を特定するコードを入れるレジスタが必要である。ただし、上記のような2つのビットフィールドの論理をとる命令ではフィールド幅WDは必然的に同一であるので、レジスタは共用させることができる。

第1段には、上記ビットフィールド間演算命令において使用されるレジスタとその中に格納されるデータとの関係の一例が示されている。

第1段

R0	ソース側ビットフィールドのBA
R1	ソース側ビットフィールドのOff[
R2	フィールド幅(WD)
R3	デスティネーション側ビットフィールドのBA
R4	デスティネーション側ビットフィールドのOff[
R5	演算の種類

同図における符号BAはベースアドレス、Off[はオフセットを示す。

また、第2段には、上記レジスタR5によって指定される演算の種類の一覧表が示されている。

第 2 表

№	演算の種類	内 容
1	True	$1 \rightarrow \text{dest}$
2	False	$0 \rightarrow \text{dest}$
3	Not Dest	$\overline{\text{dest}} \rightarrow \text{dest}$
4	Dest	$\text{dest} \rightarrow \text{dest}$
5	Not Src	$\overline{\text{src}} \rightarrow \text{dest}$
6	Src	$\text{src} \rightarrow \text{dest}$
7	AND	$\text{dest. and. src} \rightarrow \text{dest}$
8	Or	$\text{dest. or. src} \rightarrow \text{dest}$
9	Xor	$\text{dest. xor. src} \rightarrow \text{dest}$
10	Not And	$\overline{\text{dest. and. src}} \rightarrow \text{dest}$
11	Not Or	$\overline{\text{dest. or. src}} \rightarrow \text{dest}$
12	And Not	$\text{dest. and. } \overline{\text{src}} \rightarrow \text{dest}$
13	Or Not	$\text{dest. or. } \overline{\text{src}} \rightarrow \text{dest}$
14	Not And Not	$\overline{\text{dest. and. } \overline{\text{src}}} \rightarrow \text{dest}$
15	Not Or Not	$\overline{\text{dest. or. } \overline{\text{src}}} \rightarrow \text{dest}$
16	Not Xor	$\overline{\text{dest. xor. src}} \rightarrow \text{dest}$

ットフィールドに入れる操作を、 Or で示される演算は、ソース側とデスティネーション側のビットフィールド内のデータの論理和をとってデスティネーション側のビットフィールドに入れる操作を、 Xor で示される演算は、ソース側とデスティネーション側のビットフィールド内のデータの排他的論理和をとってデスティネーション側のビットフィールドに入れる操作を、 Not And で示される演算は、デスティネーション側のビットフィールド内のデータの反転値とソース側ビットフィールド内データとの論理積をとってデスティネーション側のビットフィールドに入れる操作を、 Not Or で示される演算は、デスティネーション側のビットフィールド内のデータの反転値とソース側ビットフィールド内データとの論理和をとってデスティネーション側のビットフィールドに入れる操作を、 Not And Not で示される演算は、デスティネーション側のビットフィールド内のデータの反転値とソース側反転ビットフィールド内データとの論理積をとってデスティネーション側のビットフィールドに入れる操作を、 Not Or Not で示される演算は、デスティネーション側のビットフィールド内のデータの反転値とソース側反転ビットフィールド内データとの論理和をとってデスティネーション側のビットフィールドに入れる操作を、 Not Xor で示される演算は、デスティネーション側のビットフィールド内のデータの反転値とソース側ビットフィールド内データとの排他的論理和をとってデスティネーション側のビットフィールドに入れる操作をそれぞれ意味する。

同様において、 True で示される演算はデスティネーション側ビットフィールドの全ビットを "1" にする操作を、 False で示される演算はデスティネーション側ビットフィールドの全ビットを "0" にする操作を意味する。また、 Not-Dest で示される演算はデスティネーション側ビットフィールド内の全ビットのデータを反転して元のビットフィールドに入れる操作を、 Dest で示される演算はデスティネーション側ビットフィールド内のデータをそのまま元のビットフィールドに戻す操作を、 Not で示される演算はソース側ビットフィールド内の全ビットのデータを反転してデスティネーション側ビットフィールドに入れる操作を、そして、 Src で示される演算はソース側ビットフィールド内のデータをデスティネーション側ビットフィールドに入れる操作を意味する。

さらに、 AND で示される演算は、ソース側とデスティネーション側のビットフィールド内のデータの論理積をとってデスティネーション側のビット

ットフィールドに入れる操作を、 Or Not で示される演算は、デスティネーション側のビットフィールド内のデータとソース側ビットフィールド内データの反転値との論理和をとってデスティネーション側のビットフィールドに入れる操作を、 Not And Not で示される演算は、デスティネーション側のビットフィールド内のデータの反転値とソース側ビットフィールド内のデータ反転値との論理積をとってデスティネーション側のビットフィールドに入れる操作を、 Not Or Not で示される演算は、デスティネーション側のビットフィールド内のデータの反転値とソース側反転ビットフィールド内データとの論理和をとってデスティネーション側のビットフィールドに入れる操作を、 Not Xor で示される演算は、デスティネーション側のビットフィールド内のデータの反転値とソース側ビットフィールド内データとの排他的論理和をとってデスティネーション側のビットフィールドに入れる操作をそれぞれ意味する。

上記各種演算は例えばレジスタ R5 の下位 4 ビットによって識別させることができる。

上記のようなビットフィールド間演算命令を使用すると、演算の種類がオペランドの一つとして与えられるため、メモリ内のデータを変更するかメモリからロードするデータを変えるだけでプログラム実行中に演算の種類をダイナミックに変更することができる。ただし、この実施例のビットフィールド間演算命令を実行する前に、予めオペランドとして与えられるベースアドレスやオフセットおよび演算の種類を示すコードを所定のレジスタ (R0 ~ R5) に入れておいてやる必要がある。

第3表に、上記ビットフィールド間演算命令 (BVMAP と略す) を使用したグラフィック表示用のプログラムの一例が示されている。

第3表

```
LOOP    MOVE (R10)+, R0
        MOVE (R11)+, R1
```

理を行なった結果が画像として表示されるようになる。

上記プログラムでは、ラインごとにレジスタ R5 の内容を変えることにより演算の種類を変更しているが、本発明はこの実施例に限定されるものではない。例えば上記プログラムの前に実行されるプログラムによって得られた結果をレジスタ R5 に格納しておき、レジスタ R5 の内容を更新せずに上記プログラムを実行することもできる。これにより、プログラム実行中に演算の種類をダイナミックに変更することができる。

第7図にはビットフィールド間演算命令 BVMAP のフローチャートが示されている。ステップ S1において、レジスタ R0, R1 及び R2 の内容を用いてソース側ビットフィールドがフェッチされる。ステップ S2においてレジスタ R2, R3 及び R4 の内容を用いてデスティネーション側ビットフィールドがフェッチされる。ステップ S3において、レジスタ R5 の内容を用いて演算が実行される。ステップ S4において終了条件を判

```
MOVE (R12)+, R2
MOVE (R13)+, R3
MOVE (R14)+, R4
MOVE (R15)+, R5
BVMAP
SUB LINE, -1
BNE LOOP
```

上記プログラムは、ラインごとにポストインクリメントによってレジスタ R0 ~ R5 の内容を変えながら BVMAP で示されるビットフィールド間演算命令を繰返し実行することを処理の内容としている。例えば MOVE (R10)+, R0 は、レジスタ R0 の内容を更新してレジスタ R0 に格納する命令である。また SUB LINE, -1 は、繰り返し数から 1 を引く命令である。

従って、上記プログラムにおいてレジスタ R5 に格納される演算種類を例えば一回ごとに変えながら繰返し実行してやれば、表示画面上において一行ごとに演算内容の異なるビットフィールド処

理を行なった結果が画像として表示されるようになる。

定し、終了条件が一致すれば命令を終了し、不一致ならば上記ステップ 1 に戻る。一度にフェッチできるデータのビット数はマイクロプロセッサのデータバス長によって定まる。従って、ビットフィールドの全てをフェッチし、これに基づいて演算を行うためには上記ステップ S1 ~ S3 を複数回繰り返すことが必要な場合がある。

なお、上記実施例では、ビットフィールド命令として 2 つのビットフィールドのデータ同士の操作に関する命令を例に挙げて説明したが、グラフィック処理に適したビットフィールド命令としては、その他に例えばベースアドレスとオフセット及びフィールド幅で指定されたビットフィールドに対して任意のレジスタのビットパターンを繰返し格納させるような命令が考えられる。この命令を使用すると画面上の任意の領域を任意のパターン（模様を構成する基本图形）で埋めて行くような一種の塗りつぶし処理が容易に行なえるようになる。

第3図には、上記実施例のビットフィールド命

令を有する命令体系によって動作するマイクロプロセッサのハードウェア構成の一例が示されている。

この実施例のマイクロプロセッサは、マイクロプログラム制御方式の制御部を備えている。すなわち、マイクロプログラムが格納されたマイクロROM(リード・オンリ・メモリ)2が設けられている。マイクロROM2は、マイクロアドレス発生回路5によってアクセスされ、マイクロプログラムを構成するマイクロ命令を順次出力する。

マイクロアドレス発生回路5は、命令レジスター3にフェッチされたマイクロ命令のコードを、命令デコード4でデコードした信号が供給される。マイクロアドレス発生回路5はこの信号に基づいて対応するマイクロアドレスを形成し、マイクロROM2に供給する。これによって、そのマイクロ命令を実行する一連のマイクロ命令群の最初の命令が読み出される。このマイクロ命令コードに

録される。これによって、プログラムの読み込みが高速化される。

なお、上記実施例では、一例としてグラフィック処理に適したピットフィールド命令に適用したものについて説明したが、それ以外の演算命令に適用することができる。

また、上記実施例では、オペランドで指定する演算の種類がアンド(AND)やオア(OR)などの論理演算に限定されているが、算術演算を行なう命令についても同様にオペレーションコードを同一にし、かつ演算の種類をオペランドで指定するようにしてもよい。

第4図は、第3図に示す実行ユニット6の内部ブロック図を示している。

第4図の実行ユニットにおいて、回路符号CB8で示されているのは、オフセット値やフィールド幅等の拡張データをラッピングするためのレジスタ、D0Rはメモリヘストアするデータをラッピングするためのデータ・アウトプット・レジスタ、DI8はメモリから読み出されたデータをラッピングするデ

ィジタルデータバッファ、演算処理ユニット等からなる実行ユニット6等に対する制御信号が形成される。この実行ユニット6内に上記実施例で使用された汎用レジスターR0～R15が含まれている。

マイクロ命令に対応する一連のマイクロ命令群のうち2番目以降のマイクロ命令の読み出しは、直前に読み出されたマイクロ命令のネクストアドレスフィールドのコードがマイクロROM2に供給されることより、直前のマイクロ命令内のネクストアドレスとマイクロアドレス発生回路5からのアドレスとに基づいて行われる。このようにして、一連のマイクロ命令が読み出されて形成された制御信号によって実行ユニット6が制御され、マイクロ命令が実行される。

この実施例では、特に制限されないが、バッファ記憶方式が採用されており、マイクロプロセッサLSI内にキャッシュメモリ7が設けられ、外部メモリ8内でのデータのうちアクセス頻度の高いプログラムデータがキャッシュメモリ7内に登

録される。データ・インプット・レジスタ、ALNは入出力されるデータを整列させるアライナで、このアライナALNはデータI/Oインターフェース(図示省略)を介して外部のデータバスに接続される。

また、回路符号BSFで示されるのは、32ビットずつ同時に入力された64ビットデータから任意の32ビットを抽出するためのパレルシフタで、このパレルシフタBSFは、0のようを定数を直接入力できるように構成されている。BCNTはパレルシフタBSFに対して抽出する位置を指定するパレルシフタ・カウンタ、BSFOはパレルシフタBSFの出力をラッピングするレジスタである。また、FBはデータを入力させることにより上位27ビットをマスクして出力する等の働きをするファンクション・ブロック、FB0はファンクション・ブロックFBの出力をラッピングするレジスタである。

さらに、回路符号AUで示されるのは実効アドレスを計算するためのアドレス演算ユニットで、このアドレス演算ユニットAUは、0のようを定

数を直接入力できるように構成されている。AU0はこのアドレス演算ユニットAUの出力をラッテするレジスタ、SPTはアドレス演算ユニットAUで演算される前のデータのシフトを行なうシフタ、AOTは演算結果の入っている上記レジスタAU0の値を後述のテンボラリレジスタDTE0～DTE3へ移す際に一時的に保持するラッテ回路、AORは同じくレジスタAU0のアドレス値を外部へ出力する際に一時的に保持するアドレス・アウトプット・レジスタで、このレジスタAORはアドレスI/Oインターフェース(図示省略)を介して外部のアドレスバスに接続される。

一方、回路符号ALUで示されているのは、加算、減算等基本的な算術演算や論理演算を行なう演算論理ユニット、ALU0は演算論理ユニットALUでの演算結果をラッテするレジスタ、また、DTE0～DTE3で示されるのは、テンボラリ値をラッテする外部から見えない(ユーザーに開放されていない)レジスタ群、R0, R1, ……R16はユーザーに開放されている汎用レジスタ群で

の信号である。制御信号I2は、インバータ回路INVの動作を制御する信号であり、入力信号を反転して出力するか、反転せずに出力するかの選択をする信号である。制御信号I3は演算論理ユニットALUの演算機能の選択をする信号である。演算論理ユニットALUは、論理積(AND)、論理和(OR)又は排他的論理和(XOR)等の演算機能を有しており、いずれかの機能が制御信号I3によって選択されることになる。制御信号I5はレジスタALU0にラッテされたデータをBCバスに送出するか又は演算論理ユニットALUの入力側に帰還するかの選択を行う信号である。制御信号I4は、上記入力側に帰還されたデータ又はオールゼロ(0)の一方を選択するための信号である。制御信号I1によって選択されたデータはインバータ回路INVを介して演算論理ユニットALUの一方の入力データとされ、制御信号I4によって選択されたデータは演算論理ユニットALUの他方の入力データとされる。この演算論理ユニットALUの動作は2段階に分けられる。例

あり、上記各種レジスタやラッテ回路、演算器等は、4種類のバスECB, BA, BB, BCを介して相互に接続され、マイクロROMからなる制御部より供給される制御信号によって、シーケンシャルに動作され、対応するマクロ命令が実行される。

本発明によれば上記演算論理ユニットALU等は汎用レジスタの内容、例えばレジスタR5の内容によって制御可能とされる。

第5図は、第4図に示す演算論理ユニットALU等及びこれらを制御するためのレジスタR5との関係を示している。

汎用レジスタR5の内容は、特に限定されないが一般他のレジスタINFRにストアされ、このレジスタINFRから制御信号I1～I5が出力される。レジスタINFRは第4図において省略されているが、上記テンボラリレジスタDTE0等と同様のレジスタとされ、実行ユニット内に設けられる。制御信号I1は、BBバス上のデータ又はオールゼロ(0)のデータの一方を選択するため

第 4 表

#	I1による選択	I2による選択	I3による選択
1	O	非反転	OR
2	O	非反転	OR
3	O	非反転	OR
4	O	非反転	OR
5	src	反転	OR
6	src	非反転	OR
7	src	非反転	OR
8	src	非反転	OR
9	src	非反転	OR
10	src	非反転	OR
11	src	非反転	OR
12	src	反転	OR
13	src	反転	OR
14	src	反転	OR
15	src	反転	OR
16	src	非反転	OR

第 5 表

番	I 1Cによる選択	I 2Cによる選択	I 3による選択
1	O	反転	OR
2	O	非反転	OR
3	dest	反転	OR
4	dest	非反転	OR
5	O	非反転	OR
6	O	非反転	OR
7	dest	非反転	AND
8	dest	非反転	OR
9	dest	非反転	XOR
10	dest	反転	AND
11	dest	反転	OR
12	dest	非反転	AND
13	dest	非反転	OR
14	dest	反転	AND
15	dest	反転	OR
16	dest	反転	XOR

ンパート回路 INV で反転され、オールワン(1)にされて演算論理ユニット ALU の一方の入力データとされる。また演算論理ユニット ALU の他方の入力データは上記演算結果(オールゼロ)とされる。演算論理ユニット ALU の演算機能は、制御信号 I 3 によって論理和(OR)とされるから、その演算結果はオールワン(1)とされる。このデータをデスティネーション側ビットフィールドにストアすることによって、上記演算 $1 \rightarrow \text{dest}$ の実行が終了する。

また、第 10 番目の演算 $\overline{\text{dest}} \cdot \text{AND} \cdot \text{src} \rightarrow \text{dest}$ を実行する場合には、まず、第 6 図(A)において、第 4 表の第 10 番目に記載された制御が行なわれる。すなわち、制御信号 I 1 によって、BB 上のデータ(この場合ソース側ビットフィールドの値) src が選択され、これがインパート回路 IND で反転されずに演算論理ユニット ALU の一方の入力データとされる。また演算論理ユニット ALU の他方の入力データはオールゼロ(0)とされる。

えば、前記第 2 表に記載した各演算を行なう場合において、第 1 段階の動作状態が第 6 図(A)に示され、第 2 段階の動作状態が第 6 図(B)に示されている。

上記第 4 表は、上記第 6 図(A)に示す動作状態をさらに詳細に説明するための表であり、第 5 表は上記第 6 図(B)に示す動作状態をさらに詳細に説明するための表である。例えば第 2 表に示す第 1 番目の演算 $1 \rightarrow \text{dest}$ を実行する場合には、まず、第 6 図(A)において、第 4 表の第 1 番目に記載された制御が行なわれる。すなわち、制御信号 I 1C によってオールゼロ(0)が選択され、これがインパート回路 IND で反転されずに演算論理ユニット ALU の一方の入力データとされる。また演算論理ユニット ALU の他方の入力データはオールゼロ(0)とされる。この演算論理ユニット ALU の演算機能は、制御信号 I 3 によって論理和(OR)とされるから、その演算結果はオールゼロ(0)とされる。次に第 6 図(B)において、第 5 表の第 1 番目に記載された制御が行なわれる。すなわち制御信号 I 1 によってオールゼロ(0)が選択され、これがイ

この演算論理ユニット ALU の演算機能は、制御信号 I 3 によって論理和(OR)とされるから、その演算結果はソース側ビットフィールドの値(src)とされる。次に第 6 図(B)において、第 5 表の第 10 番目に記載された制御が行なわれる。すなわち制御信号 I 1 によって BB BUS 上のデータ(この場合デスティネーション側ビットフィールドの値) dest が選択され、これがインパート回路 INV で反転され(dest) 演算論理ユニット ALU の一方の入力データとされる。また演算論理ユニット ALU の他方の入力データは上記演算結果(src)とされる。演算論理ユニット ALU の演算機能は、制御信号 I 3 によって論理積(AND)とされるから、その演算結果は dest · AND · src とされる。このデータをデスティネーション側ビットフィールドにストアすることによって、上記演算 $\overline{\text{dest}} \cdot \text{AND} \cdot \text{src} \rightarrow \text{dest}$ の実行が終了する。

上記実施例では、レジスタ R 5 の内容によって、演算論理ユニット ALU 等が直接制御されている

が、本発明の適用に際して上記実施例に限定されるものではない。すなわち汎用レジスタ群R₀～R₁₁の内容によって演算論理ユニットA LU等が間接的に制御されるものであってもよい。例えば、レジスタR₅の内容を第3図に示す命令コード4等に供給することにより、マイクロROM2から上記制御信号I₁～I₅等を得ることもできる。また上記実施例では、制御信号I₁はBBバス上のデータを選択するか否かの選択信号とされているが、これに限定されない。例えば、BBバス上にデータを供給するための供給源等を、この制御信号I₁で制御することもできる。なお、BBバス上への、ソース側ビットフィールドの値src又はデスティネーション側ビットフィールドの値destの送出は、ペルシフタBSFの出力ラッチレジスタBSFOからなされる。

第8図は、第7図に示すフローチャートにおけるステップS1をさらに詳細に説明するための実行手順が示されている。ステップS1は、ステップS01からS12によって構成される。なお、

スト指令が与えられる。これによって、レジスタAOR内のアドレス値がI/Oインターフェースを介して外部アドレスバス上に出力され、外部のメモリがアクセスされてその内容がデータバス上に出力される。そして、メモリから読み出されたデータすなわちオフセット値OffsetがI/Oインターフェースによりフェッチされる。

第3のステップS03では、I/Oインターフェースより出力される信号に基づいてフェッチしたデータが確定しているか確認される。これによって、データはデータ・インプット・レジスタDIRに取り込まれる。これとともに、ステップS03では、レジスタR₀の値すなわちソース・ベースアドレスBADがバスBAまたはBBを介して、また定数入力機能によって0が、アドレス演算ユニットAUにそれぞれ入力され、その加算結果がレジスタAUOに格納される。

それから、第4のステップS04で、上記レジスタAUOの値(ソース・ベースアドレス)がレジスタAORへ転送され、更にバスBCを介して

第4図に示す汎用レジスタ群R₀～R₁₁の中で、符号R_a、R_b、R_x、R_yが付記されているレジスタはそれぞれソース・ベースアドレス、デスティネーション・アドレス、オフセット値・アドレス及びビットフィールド幅を入れるために使用されているレジスタであることを示している。こR_a、R_b、R_x、R_yは、汎用レジスタR₀～R₁₁の任意のレジスタ番号をそれぞれの使用目的のために指定して使用することができる。

第1のステップS01では、レジスタRx内の値すなわちオフセット値の入っている位置を示すアドレスがバスBAまたはBBを介して、また定数入力機能によって0がアドレス演算ユニットAUにそれぞれ入力され、その加算結果がレジスタAUOに格納される。

第2のステップS02では、ステップS01でレジスタAUOに格納されたアドレス値(オフセット値アドレス)がレジスタAORへ転送されるとともに、I/Oインターフェースに対して外部データバス上のデータをフェッチするようにリクエ

テンポラリレジスタDTEOに転送される。

なお、他の処理との関係でアドレス演算ユニットAUの演算結果がレジスタAUOからAORへ転送されるとき、自動的にレジスタAOTKも転送されるようになっている。ここでは、レジスタAOTへの転送は特別の意味も持っていない。これと並行して、アドレス演算ユニットAUには、定数入力機能により0が入力されるとともに、バスBBを介してデータ・インプット・レジスタDIRの内容(オフセット値)が符号拡張されて入力され、演算結果がレジスタAUOに格納される。さらに、レジスタRyからファンクションブロックFBに對して保持値すなわちビットフィールド幅WBがバスBAを介して供給され、ファンクションブロックFBで下位5ビットを除く上位27ビットがマスクされ、結果がレジスタFB0に格納される。ビットフィールド幅の下位5ビットのみ抽出することは、数学的に表現するとビットフィールド幅を数「32」で割った余りを求めることと同様である。以下、このビットフィールド

幅の下位 5 ビットを端数 WD_5 と記す。ここで端数 WD_5 を求めるのは、後のステップ S08 でのバウンダリ渡りの判定に使用するためである。

次に、ステップ S06 でレジスタ AU0 の値すなわちオフセット値 Off_1 をバス BC を介してテンポラリレジスタ DTE1 へ転送する。これとともに、アドレス演算ユニット AU に対してレジスタ AU0 の値（オフセット値）とテンポラリレジスタ DTE0 の値すなわちソース・ベースアドレス BAD をシフト SFT で上位側へ 3 ビットシフトした値とが供給され、その加算結果がレジスタ AU0 に格納される。ソース・ベースアドレス BAD を上位側へ 3 ビットシフトするのは、メモリ空間をバイト単位で区切って指示できるようにされたベースアドレス BAD を、ビット単位でメモリ空間内での位置を指示できるように拡張するためである。従って、このときレジスタ AU0 に入っているのは、求めるビットフィールドのアドレス 0 番地からのビット数で表した距離である。この距離を L と記す。

た、アドレス演算ユニット AU での加算結果の下位 2 ビットをマスクしているのは、対象となるビットフィールド全体もしくはその先頭部分を含む 32 ビットのワードのアドレスを得るためにある。

第 7 のステップ S07 では、上記のようにして得られたレジスタ AU0 内のワードアドレスがアドレス・アウトプット・レジスタ AOR に転送されて I/O インタフェースを介して外部へ出力されるとともに、I/O インタフェースに対しては外部データバス上のデータのフェッチを要求する指令がなされる。これによって、求めるビットフィールドの先頭部分を含むワードのメモリからのフェッチが開始される。これと並行して、レジスタ AU0 に保持されているアドレスがバス BC を介してテンポラリレジスタ DTE2 に転送される。また、ファンクションブロック FBK は、ステップ S05 で得られたビットフィールドのアドレス 0 からのビット位置を示す値 L が、レジスタ AOT からバス BA を介して供給され、その結果がレジスタ FBO に格納される。これによって、ステ

第 6 のステップ S06 では、上記レジスタ AU0 の値すなわちベースアドレスを上位側へ 3 ビットシフトした値にオフセット値 Off_1 を加えたものを、レジスタ AOT に転送する。一方、アドレス演算ユニット AU には、バス BA を介してテンポラリレジスタ DTE0 からソース・ベースアドレス BAD が入力され、またバス BB を介してテンポラリレジスタ DTE1 から転送されたオフセット値 Off_1 をシフト SFT で下位側へ 3 ビットシフトした値が入力されて加算され、その加算結果の下位 2 ビットをマスクした値がレジスタ AU0 に格納される。これとともに、レジスタ DBO 内の値 WD_5 すなわちビットフィールド幅の下位 5 ビットがバス BC を介してテンポラリレジスタ DTE3 に転送される。

上記の場合、アドレス演算ユニット AU によって、ベースアドレスに、オフセット値を下位側へ 3 ビットだけシフトした値を加算しているのは、対象となるビットフィールドの先頭に最も近いバイト単位の実行アドレスを求めるためである。ま

た、ステップ S06 で得られた求めるビットフィールドの先頭部分を含むワードアドレス（オフセットが 31 以下のときはベースアドレスと一致する）からのビットフィールドの先頭位置 Off_1 。（これも一つのオフセット値であり、以下 2 次オフセットと称する）がレジスタ FBO に保持されることになる。

既に、ステップ S08 では、上記レジスタ DBO 内の値 Off_1 。（2 次オフセット）がバス BC を介してテンポラリレジスタ DTE2 に転送される。これとともに、演算論理ユニット ALU に対して、バス BA および BB を介して、テンポラリレジスタ DTE3 内の値 WD_5 。（ビットフィールド幅の下位 5 ビット）とレジスタ DBO 内の値 Off_1 が供給されて加算され、その結果がレジスタ ALU0 に格納される。これとともに、レジスタ CBS に対し、制御部の側から定数「33」が設定される。「33」なる数は 1 ワードのビット数「32」に「1」を加えた数である。また、I/O インタフェースからの信号に基づいてフ

ッテしたデータすなわち求めるビットフィールドの内容が確定しているか確認する。データが確認している場合、そのデータはデータ・インプット・レジスタDIRに取り込まれることになる。

次のステップS09では、I/Oインターフェースによりフェッチされた値がデータ・インプット・レジスタDIRからバスBCを介してテンボラリレジスタDTE2に転送される。これと並行して、演算論理ユニットALUではレジスタALU0の値(Off+WD)からレジスタCBSの値「33」の演算が行なわれ、その結果がレジスタALU0に格納される。ここで、この演算結果が「正」ならばビットフィールドが2つのワードにまたがっていることを意味し、「負」ならば1ワード内に納まっていることを意味する。

また、ステップS09では、次のステップにおいてパレルシフタBSFで行なわれるビットシフト処理のシフト方向とシフト量の指定を行なう。具体的にはパレルシフタ・カウンタBCNTに対して右方向シフトの指示が与えられるとともに、

リジスタDTE3内のビットフィールド幅WDがシフト量として供給される。

そして、ステップS11では、パレルシフタBSFに対して、レジスタBSFOの値と「0」が入力され、指定された方向とシフト量に従ったシフト動作が実行され、結果がレジスタBSFOに格納される。レジスタBSFO内に入っていたビットフィールドの内容がフィールド幅WDの分だけ右シフトされると、32ビットのレジスタBSFO内には第11図に示すようにフェッチされたビットフィールドの内容が右端に寄った状態すなわちレジスタの下位側から順に詰まつた所望の状態で格納されるようになる。

このようにして、得られたビットフィールドの内容が、次のステップS12において、レジスタBSFOからバスBCを介して汎用レジスタの一つ目に格納される。

さらに、ステップS09において、Off+WDとの和と定数「33」との演算結果が「正」となって、ビットフィールドがパウンダリ渡りを

バスBAを介してレジスタFB0内の値Off+WDとしてパレルシフタ・カウンタBCNTに供給される。

そして、ステップS10において、パレルシフタBSFに對してテンボラリレジスタDTE2の値すなわちメモリからフェッチされたビットフィールドの内容がバスBBを介して供給されるとともに、定数入力機能によって0が入力されてパレルシフタ・カウンタBCNTの指示に従ったシフトが実行され、その結果がレジスタBSFOに格納される。これによって、32ビットのレジスタBSFO内には、第8図に示すようにフェッチされたビットフィールドの内容が左端に寄った状態つまりレジスタの上位ビット側から順に詰まつた状態で格納される。これと並列して、ステップS10では、次のステップにおいてパレルシフタBSFで行なわれるシフト動作の方向指示と、シフト量の指定が行なわれる。すなわち、パレルシフタ・カウンタBCNTに對し、右方向シフトの指示が与えられ、かつバスBAを介してテンボラ

していると判定された場合には、ステップS12から再びステップS08に戻って上記手順を繰り返すことにより、複数のワードにまたがっているビットフィールドのすべての内容が読み出される。

第10図に、上記マイクロフローにおけるオフセット値Off+WDとビットフィールド幅WDとの関係を示しておく。

なお、第8図のマイクロフローに従った制限なしビットフィールド命令の実行手順では、パウンダリ渡りの判定を、2次オフセットOff+WDとビットフィールドの端数WDとの和のから数「33」を引いた結果が「正」か「負」かで行なっている。本来、パウンダリ渡りの判定は1次オフセットOff+WDとビットフィールド幅WDとから判定すべきであり、そのようなマイクロフローを記述することも可能である。ただし、1次オフセットOff+WDをビットフィールド幅WDを加えてそれをベースアドレスBADから32ビットずつ区切っていくことでパウンダリ渡りが生じているか否か判定し

た場合と、実施例のように2次オフセットOff。にビットフィールドの端数WD。を加え、それが数「32」を超えたか否かでバウンダリ越りの判定を行なった場合とで全く同じ結果が得られることは、第7図からも明らかである。

以上本発明者によってなされた発明を実施例に基づき具体的に説明したが、本発明は上記実施例に限定されるものではなく、その要旨を逸脱しない範囲で種々変更可能であることはいうまでもない。例えば上記実施例ではビットフィールドのベースアドレスやオフセット、フィールド幅及び演算の種類をオペランドで与えるようにしているが、オペランドの代わりに実効アドレス部で与えるようにしてよい。

また、オペランドで演算の種類を指定するようにした本発明に係る演算命令は、従来の固定された演算の代わりに設けててもよいが、併用して設けるようにしてもよい。

以上の説明では主として本発明者によってなされた発明をその背景となつた利用分野であるマイ

クロプロセッサの命令形式に適用した場合について説明したが、この発明はそれに限定されるものではなく、計算機やミニコン等プログラム制御方式のデータ処理システム一般の命令形式に利用することができる。

(発明の効果)

本願において開示される発明のうち代表的なものによって得られる効果を簡単に説明すれば下記のとおりである。

すなわち、プログラムに柔軟性を持たせ、例えばグラフィック処理用のプログラムの開発が容易に行なえるようになる。

4. 図面の簡単な説明

第1図及び第2図は本発明の適用の対象となつたビットフィールド間演算命令におけるビットフィールドの構成例を示す説明図。

第3図は本発明に係るビットフィールド命令を実行するマイクロプロセッサの構成例を示すブロック図。

第4図は、第3図に示す実行ユニットの内部構

成を示すブロック図。

第5図、第6図(A)及び第6図(B)は第4図に示す演算論理ユニットAUU。

第7図は本発明が適用されたビットフィールド間演算命令の実行シーケンスを示すフローチャート。

第8図は第7図におけるステップS1をさらに詳細に説明するための実行シーケンス。

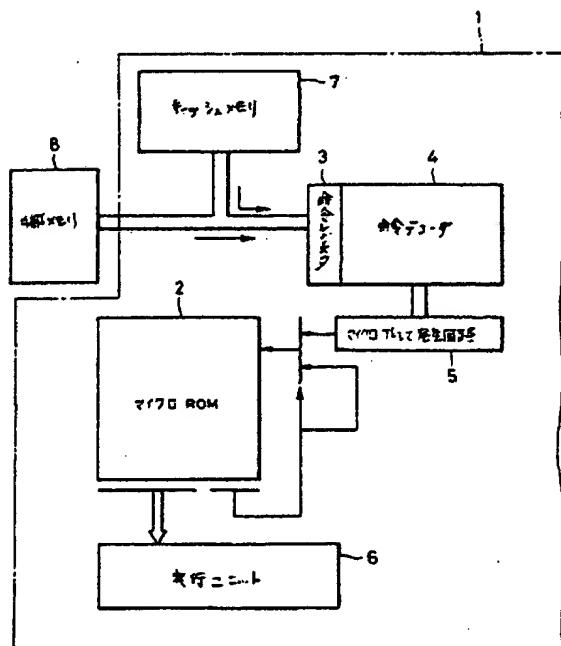
第9図(A), (B)はこの発明が適用される命令のフォーマットの実施例。

第10図及び第11図は、第8図に示す実行シーケンス中の動作の説明図である。

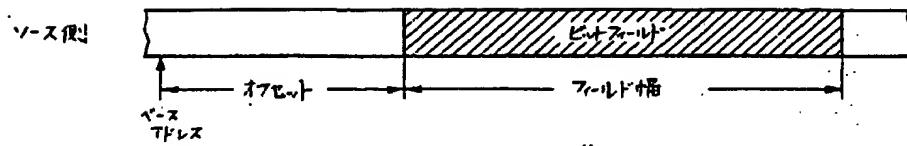
DOR…データ・アウトプット・レジスタ、DIR…データ・インプット・レジスタ、ALN…アライナ、BSF…パレルシフタ、BCNT…パレルシフタ・カウンタ、FB…ファンクションブロック、AU…アドレス演算ユニット、SFT…シフタ、AUO…ラッチ回路、AOH…アドレス・アウトプット・レジスタ、ALU…演算論理ユニット、INV…インバータ回路

代理人弁理士 小川勝男

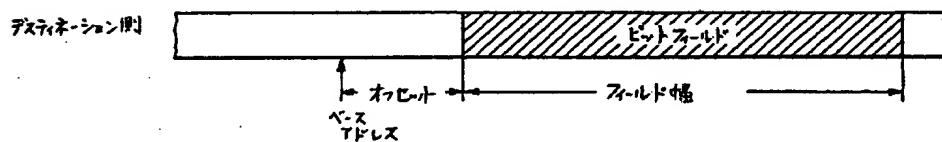
第3図



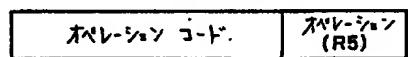
第1回



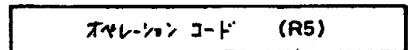
第2回



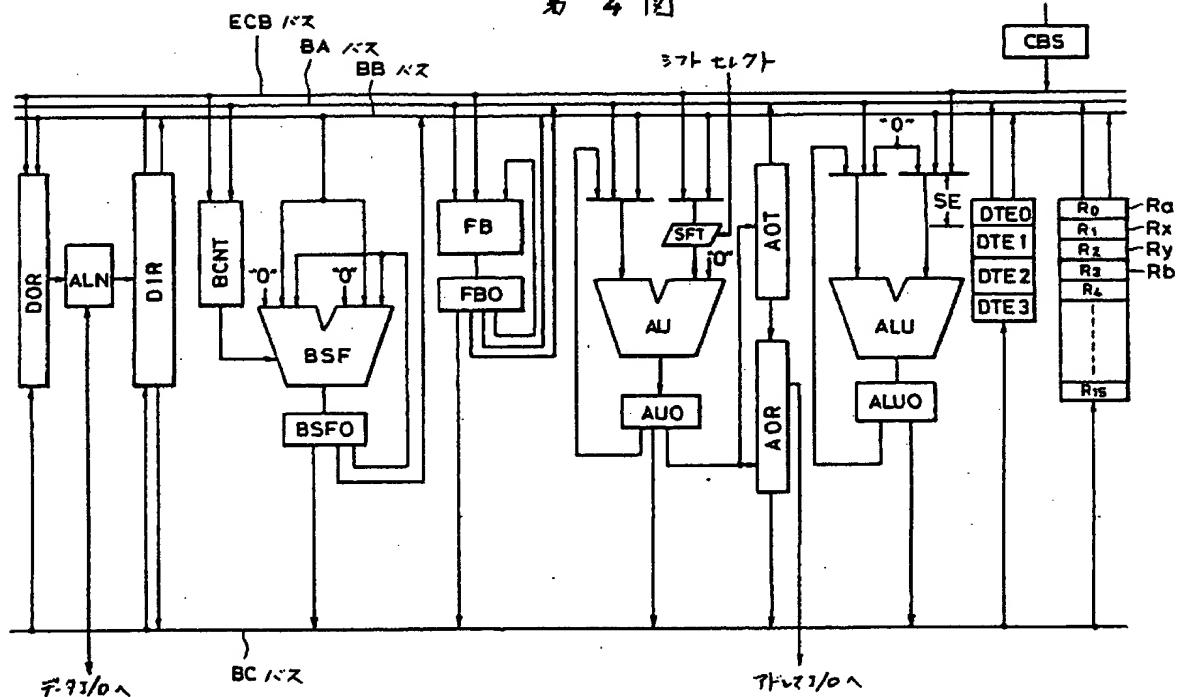
第 9(A) 図



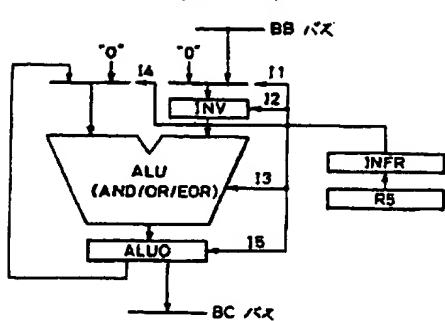
第 9(B) 図



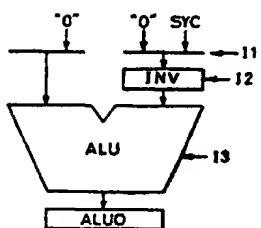
第四回



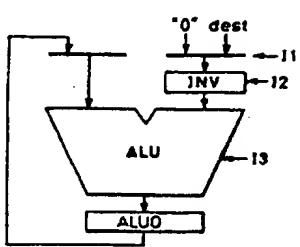
第 5 図



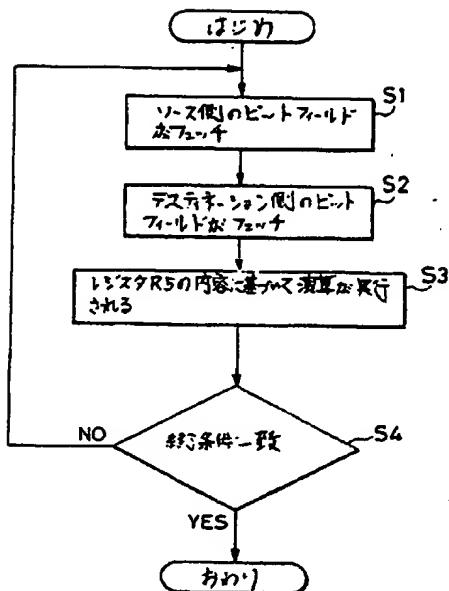
第 6 図 (A)



第 6 四



第 7 圖



圖版 8

新編和漢書

卷之三

にデータが複数しているかどうかを確認する。

卷之三

このように、各部品の機能を明確に定義することで、各部品の動作を独立して実現することができる。また、各部品の動作が独立して実現されるため、各部品の動作を変更する場合、他の部品の動作に影響を与えることはない。

車両の運転席側の窓を脱着式バスを介してDITEへ搬送する。車両跡と脱着バスを介して入力される

この問題の解き方を示す。まずバスを介して入力されるデータの構造を記述する。この問題の下二段

町の話をビデオで紹介して町PRに貢献する。

マスクし、結果を正確に推測する。

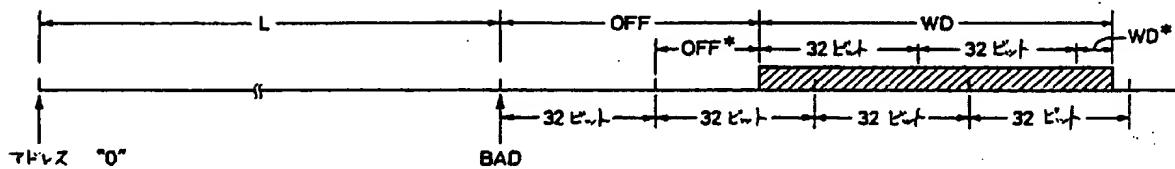
車両の運転をバスを介してCITEへ送信する。ここでバスを介して入力される車両の位置と座標をもとにデータを計算し、結果を画面に表示する。CITEへ定められた33項目を記述する。100にフェッチしたデータが定義しているかどうかを確認する。

モモリファエチによって構成したDIFの組をEIPを介してDTEへ転送する。ここでDIFの組とEIPを介して入力されるDSの定義値とを比較して、結果を山形へ送信する。結果に右方指示表示を示すと同時にEIPの組を入力する。

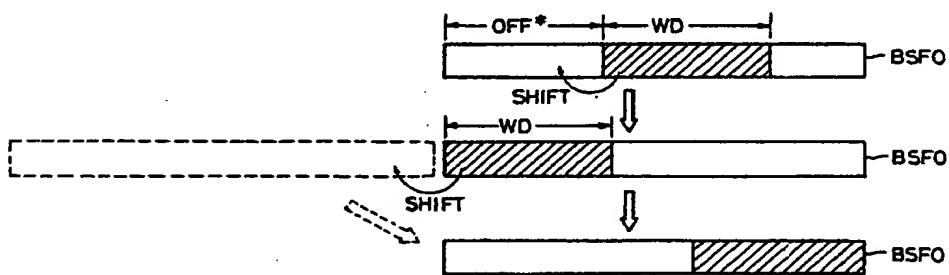
画面にパスを入力してログインをクリック。結果を表示する。また右側に表示を示すと同時にパスを入力する。

アカウントの登録が完了する。

第10図



第11図



第1頁の続き

⑤Int.Cl.⁴ 識別記号 廈内整理番号
G 06 F 15/72 400 6615-5B

⑥発明者 長谷川 淳 東京都小平市上水本町1479番地 日立マイクロコンピュータエンジニアリング株式会社内

⑦発明者 岩崎 一彦 東京都国分寺市東恋ヶ窪1丁目280番地 株式会社日立製作所中央研究所内

特開平1-156824(16)

手取種付正書(方式)
昭和 62年 3月 11日

特許庁長官 謹

事件の表示

昭和 62年 特許第 314063号

発明の名称

マイクロプロセッサ

補正をする者

事件との関係 特許出願人
名 称 (510) 株式会社 日立製作所
(他 2名)

代 理 人

居 所 〒100 東京都千代田区丸の内一丁目5番1号
株式会社日立製作所内
電話 東京 212-5111(大代表)

氏 名 (6650) 弁理士 小川勝男



補正命令の日付 昭和 63年 2月 23日

補正の対象
図面



補正の内容

明細書に添付した図面の第8図を別紙のとおり補正する。